

Horn Linear Logic and Minsky Machines

(adapted, reshuffled, self-contained, fully detailed,
cleansed, simplified, with pictures by myself, etc.)

Max Kanovich

University College London, UK and
National Research University Higher School of Economics

Twenty Years After

Contents

1	From linear logic to Horn Linear Logic HLL	2
1.1	Language: Horn sequents	2
1.2	LL rules used in cut-free derivations for Horn sequents	3
1.3	The Inference Rules of Horn Linear Logic	3
1.4	The proof of Theorem 1.1: From LL derivations to HLL derivations.	4
2	From HLL to tree-like Horn programs	6
3	FYI: Encoding Minsky Machines	10
3.1	The Main Encoding	11
3.2	FYI: From computations to tree-like Horn programs	12
4	From tree-like Horn programs to Minsky computations	15

List of Tables

1	The linear logic rules we use for Horn sequents. Here A is a Horn implication or \oplus -Horn implication	3
2	Horn Linear Logic HLL . Both \otimes and \oplus are assumed to be commutative and associative. . .	4

List of Figures

1	Elementary Horn Programs.	8
2	The “Frame property”	8
3	Strong Forking. An \oplus -Horn implication $(X \multimap (Y_1 \oplus Y_2))$ as Non-deterministic choice	9
4	Cut as Composition of programs	10
5	The correspondence: Computation (a) — Horn Program (b).	13
6	The <i>assignment operation</i> correspondence: (a) – (b).	16
7	The <i>ZERO-test</i> correspondence: (a) – (b).	17

Here we give a detailed proof for the crucial point in our Minsky machine simulation:

Theorem 4.1 *Any linear logic derivation for a Horn sequent of the form*

$$(l_1 \otimes (r_1^{k_1} \otimes r_2^{k_2} \otimes \cdots \otimes r_n^{k_n})), !\Phi_M, !\mathcal{K} \vdash l_0$$

can be transformed into a Minsky computation leading from an initial configuration of the form

$$(L_1, k_1, k_2, \dots, k_n)$$

to the halting configuration

$$(L_0, 0, 0, \dots, 0).$$

For the sake of perspicuity I include the information about the main encoding. In particular, this specifies what kind of Horn programs of a simple branching structure we are actually dealing with within the framework of our particular encoding.

Among other things, the presentation advantage of the 3-step program is that the non-trivial tricky points are distributed between the independent parts each of which we justify following its own intrinsic methodology (to say nothing of the induction used in the opposite directions):

- (1) From LL to HLL - we use purely proof-theoretic arguments.
- (2) From HLL to Horn programs - we translate trees (HLL derivations) into another trees (Horn programs) of the same shape, almost.
- (3) From Horn programs to Minsky computations - we use purely computational arguments.

Since the unavoidable implication of the 3-step program is undecidability of full linear logic, I would highly appreciate your comments on which issues looked suspicious to you to be addressed to with further detalization.

1 From linear logic to Horn Linear Logic HLL

We start from the purely proof-theoretic part:

Theorem 1.1 *Any cut-free derivation for a Horn sequent of the form*

$$W, \Gamma, !\Delta \vdash Z$$

*can be transformed into a derivation in Horn Linear Logic, **HLL**.*

1.1 Language: Horn sequents

The connectives \otimes and \oplus are assumed to be commutative and associative.

Here we confine ourselves to *Horn-like sequents* introduced in the following way.

Definition 1.1 The tensor product of a *positive number* of positive literals is called a **simple product**. A single literal q is also called a **simple product**.

Definition 1.2 We will use a natural isomorphism between non-empty finite multisets over positive literals and simple products: A multiset $\{q_1, q_2, \dots, q_k\}$ is represented by the simple product $(q_1 \otimes q_2 \otimes \cdots \otimes q_k)$, and vice versa.

Definition 1.3 We will write $X \cong Y$ to indicate that X and Y represent one and the same multiset M .

Definition 1.4 Here, and henceforth, $X, X', Y, Y_i, U, V, W, Z$, etc., stand for \otimes -products of positive literals.

Horn implications are defined as follows:

(i) A **Horn implication** is a formula of the form

$$(X \multimap Y).$$

(ii) A **\oplus -Horn implication** is a formula of the form

$$(X \multimap (Y_1 \oplus Y_2)).$$

Definition 1.5 A **Horn sequent** is defined as a sequent of the form

$$W, \Gamma, !\Delta \vdash Z$$

where the multisets Γ and Δ consist of Horn implications and \oplus -Horn implications, and W and Z are \otimes -products of positive literals.

1.2 LL rules used in cut-free derivations for Horn sequents

In Table 1 we collect all the inference rules of Linear Logic that can be used in cut-free derivations for Horn sequents.

(i) “Left rules”: **L \otimes** , **L \multimap** , **L $\multimap\oplus$** , **L \oplus** , **L $!$** , **W $!$** , **C $!$** .

(ii) “Right rules”: **R \otimes** .

The *intuitionistic shape* of the rules selected in Table 1 is caused by the fact that a sequent of the form

$$W, \Gamma, !\Delta \vdash$$

is not derivable in linear logic - simply replace all propositions with the constant **1**.

I	$\frac{}{X \vdash X}$	
L\otimes	$\frac{\Sigma, X, Y \vdash Z}{\Sigma, (X \otimes Y) \vdash Z}$	R\otimes $\frac{\Sigma_1 \vdash Z_1 \quad \Sigma_2 \vdash Z_2}{\Sigma_1, \Sigma_2 \vdash (Z_1 \otimes Z_2)}$
L\multimap	$\frac{\Sigma_1 \vdash X \quad Y, \Sigma_2 \vdash Z}{\Sigma_1, (X \multimap Y), \Sigma_2 \vdash Z}$	
L$\multimap\oplus$	$\frac{\Sigma_1 \vdash X \quad (Y_1 \oplus Y_2), \Sigma_2 \vdash Z}{\Sigma_1, (X \multimap (Y_1 \oplus Y_2)), \Sigma_2 \vdash Z}$	
L\oplus	$\frac{\Sigma, Y_1 \vdash Z \quad \Sigma, Y_2 \vdash Z}{\Sigma, (Y_1 \oplus Y_2) \vdash Z}$	
L$!$	$\frac{\Sigma, A \vdash Z}{\Sigma, !A \vdash Z}$	
W$!$	$\frac{\Sigma \vdash Z}{\Sigma, !A \vdash Z}$	C$!$ $\frac{\Sigma, !A, !A \vdash Z}{\Sigma, !A \vdash Z}$

Table 1: The linear logic rules we use for Horn sequents. Here A is a Horn implication or \oplus -Horn implication

1.3 The Inference Rules of Horn Linear Logic

The inference rules of the Horn Linear Logic **HLL** are given in Table 2.

I	$\frac{}{X \vdash X}$	L\otimes	$\frac{X, \Gamma, !\Delta \vdash Z}{Y, \Gamma, !\Delta \vdash Z} \text{ (where } X \cong Y \text{)}$
H	$\frac{}{X, (X \multimap Y) \vdash Y}$	M	$\frac{X, \Gamma, !\Delta \vdash Y}{(X \otimes V), \Gamma, !\Delta \vdash (Y \otimes V)}$
\oplus-H	$\frac{(Y_1 \otimes V), \Gamma, !\Delta \vdash Z \quad (Y_2 \otimes V), \Gamma, !\Delta \vdash Z}{(X \otimes V), \Gamma, (X \multimap (Y_1 \oplus Y_2)), !\Delta \vdash Z}$		
L!	$\frac{X, \Gamma, A, !\Delta \vdash Z}{X, \Gamma, !A, !\Delta \vdash Z}$		
W!	$\frac{X, \Gamma, !\Delta \vdash Z}{X, \Gamma, !A, !\Delta \vdash Z}$	C!	$\frac{X, \Gamma, !A, !A, !\Delta \vdash Z}{X, \Gamma, !A, !\Delta \vdash Z}$
Cut	$\frac{W, \Gamma_1, !\Delta_1 \vdash U \quad U, \Gamma_2, !\Delta_2 \vdash Z}{W, \Gamma_1, \Gamma_2, !\Delta_1, !\Delta_2 \vdash Z}$		

Table 2: Horn Linear Logic **HLL**. Both \otimes and \oplus are assumed to be commutative and associative.

1.4 The proof of Theorem 1.1: From LL derivations to HLL derivations.

Given a cut-free derivation for the sequent

$$W, \Gamma, !\Delta \vdash Z$$

by induction we will simulate each of the LL rules in Table 1 with the **HLL** rules from Table 2.

Rule **L \multimap** and the like

- (i) An (**R \otimes**)-rule of the form (here π_1 and π_2 are proofs that have been already constructed by induction with rules from Table 2):

$$\frac{\frac{\pi_1}{\Sigma_1 \vdash Z_1} \quad \frac{\pi_2}{\Sigma_2 \vdash Z_2}}{\Sigma_1, \Sigma_2 \vdash (Z_1 \otimes Z_2)} \text{ **R}\otimes**$$

is simulated with the **HLL** rules from Table 2:

$$\frac{\frac{\frac{\pi_1}{\Sigma_1 \vdash Z_1}}{Z_2, \Sigma_1 \vdash (Z_1 \otimes Z_2)} \text{ **M}** \quad \frac{\pi_2}{\Sigma_2 \vdash Z_2}}{\Sigma_1, \Sigma_2 \vdash (Z_1 \otimes Z_2)} \text{ **Cut}**$$

- (ii) An (**L \multimap**)-rule of the form (here π_1 and π_2 are proofs that have been already constructed by induction with rules from Table 2):

$$\frac{\frac{\pi_1}{\Sigma' \vdash X} \quad \frac{\pi_2}{Y, \Sigma'' \vdash Z'}}{\Sigma', (X \multimap Y), \Sigma'' \vdash Z'} \text{ **L}\multimap**$$

is simulated with the **HLL** rules from Table 2:

$$\frac{\frac{\frac{\pi_1}{\Sigma' \vdash X}}{\Sigma', (X \multimap Y) \vdash Y} \text{ **H}** \quad \frac{\pi_2}{Y, \Sigma'' \vdash Z'}}{\Sigma', (X \multimap Y), \Sigma'' \vdash Z'} \text{ **Cut}**$$

- (iii) The remaining LL rules, save for **L \oplus** and **L $\multimap\oplus$** , are processed by the same token. ■

Challenging $\mathbf{L}\oplus$ and $\mathbf{L}\multimap\oplus$

The main difficulties we meet with the rule $\mathbf{L}\oplus$ (and related to it $\mathbf{L}\multimap\oplus$) are that the positions at which these rules are applied in the given cut-free LL derivation might have happened very far from each other. First, we have to contract the distance between their positions by pushing $\mathbf{L}\oplus$ downwards in accordance with Lemma 1.1 to make the application positions of $\mathbf{L}\oplus$ and $\mathbf{L}\multimap\oplus$ *adjacent*:

Lemma 1.1 *Given a cut-free derivation with the rules from Table 1, by the appropriate ‘commuting conversions’ (see below), the left rule $\mathbf{L}\oplus$ can be pushed downwards (down to the related $\mathbf{L}\multimap\oplus$), forming a piece of the derivation where the rules $\mathbf{L}\oplus$ and $\mathbf{L}\multimap\oplus$ are sitting in the adjacent positions so that the rule $\mathbf{L}\multimap\oplus$ is applied just after the rule $\mathbf{L}\oplus$:*

$$\frac{\frac{\pi_0}{\Sigma' \vdash X} \quad \frac{\frac{\pi_1}{Y_1, \Sigma'' \vdash Z'} \quad \frac{\pi_2}{Y_2, \Sigma'' \vdash Z'}}{(Y_1 \oplus Y_2), \Sigma'' \vdash Z'} \mathbf{L}\oplus}{\Sigma', (X \multimap (Y_1 \oplus Y_2)), \Sigma'' \vdash Z'} \mathbf{L}\multimap\oplus \quad (1)$$

Proof. We consider all points of interaction between the rule $\mathbf{L}\oplus$ and other rules.

(a) A combination: “first $\mathbf{L}\oplus$, then $\mathbf{L}\multimap$,” of the form (here π_0, π_1 and π_2 are proofs):

$$\frac{\frac{\pi_1}{\Sigma', Y_1 \vdash U} \quad \frac{\pi_2}{\Sigma', Y_2 \vdash U} \mathbf{L}\oplus \quad \frac{\pi_0}{V, \Sigma'' \vdash Z'} \mathbf{L}\multimap}{\Sigma', (Y_1 \oplus Y_2) \vdash U \quad V, \Sigma'' \vdash Z'} \mathbf{L}\multimap\oplus$$

can be replaced with the following combination: “first $\mathbf{L}\multimap$, then $\mathbf{L}\oplus$.”

$$\frac{\frac{\pi_1}{\Sigma', Y_1 \vdash U} \quad \frac{\pi_0}{V, \Sigma'' \vdash Z'} \mathbf{L}\multimap \quad \frac{\pi_2}{\Sigma', Y_2 \vdash U} \quad \frac{\pi_0}{V, \Sigma'' \vdash Z'} \mathbf{L}\multimap}{\Sigma', Y_1, (U \multimap V), \Sigma'' \vdash Z' \quad \Sigma', Y_2, (U \multimap V), \Sigma'' \vdash Z'} \mathbf{L}\oplus$$

(b) A combination: “first $\mathbf{L}\oplus$, then $\mathbf{R}\otimes$,” of the form (here π_0, π_1 and π_2 are proofs):

$$\frac{\frac{\pi_1}{\Sigma', Y_1 \vdash Z'} \quad \frac{\pi_2}{\Sigma', Y_2 \vdash Z'} \mathbf{L}\oplus \quad \frac{\pi_0}{\Sigma'' \vdash Z''} \mathbf{R}\otimes}{\Sigma', (Y_1 \oplus Y_2) \vdash Z' \quad \Sigma'' \vdash Z''} \mathbf{R}\otimes$$

can be replaced with the following combination: “first $\mathbf{R}\otimes$, then $\mathbf{L}\oplus$.”

$$\frac{\frac{\pi_1}{\Sigma', Y_1 \vdash Z'} \quad \frac{\pi_0}{\Sigma'' \vdash Z''} \mathbf{R}\otimes \quad \frac{\pi_2}{\Sigma', Y_2 \vdash Z'} \quad \frac{\pi_0}{\Sigma'' \vdash Z''} \mathbf{R}\otimes}{\Sigma', Y_1, \Sigma'' \vdash (Z' \otimes Z'') \quad \Sigma', Y_2, \Sigma'' \vdash (Z' \otimes Z'')} \mathbf{L}\oplus$$

(c) The appropriate ‘commuting conversions’ for the remaining combinations: “first $\mathbf{L}\oplus$, then ...” can be constructed in a similar way. ■

Completing $\mathbf{L}\oplus$ and $\mathbf{L}\multimap\oplus$

According to Lemma 1.1, in order to complete the proof of Theorem 1.1, it suffices to take a piece of the derivation where the rules $\mathbf{L}\oplus$ and $\mathbf{L}\multimap\oplus$ are sitting in the adjacent positions so that the rule $\mathbf{L}\multimap\oplus$ is

applied **just after** the rule $\mathbf{L}\oplus$:

$$\frac{\frac{\pi_0}{\Sigma' \vdash X} \quad \frac{\frac{\pi_1}{Y_1, \Sigma'' \vdash Z'} \quad \frac{\pi_2}{Y_2, \Sigma'' \vdash Z'}}{(Y_1 \oplus Y_2), \Sigma'' \vdash Z'} \mathbf{L}\oplus}{\Sigma', (X \multimap (Y_1 \oplus Y_2)), \Sigma'' \vdash Z'} \mathbf{L}\multimap\oplus$$

and simulate it with the **HLL** rules from Table 2 as follows:¹

$$\frac{\frac{\pi_0}{\Sigma' \vdash X} \quad \frac{\frac{\pi_1}{Y_1, \Sigma'' \vdash Z'} \quad \frac{\pi_2}{Y_2, \Sigma'' \vdash Z'}}{X, (X \multimap (Y_1 \oplus Y_2)), \Sigma'' \vdash Z'} \mathbf{\oplus-H}}{\Sigma', (X \multimap (Y_1 \oplus Y_2)), \Sigma'' \vdash Z'} \mathbf{Cut}$$

■

2 From HLL to tree-like Horn programs

As computational counterparts of Horn sequents, we will consider *tree-like Horn programs* with the following peculiarities:

Definition 2.1 A **tree-like Horn program** is a rooted binary tree such that

- (a) Every edge of it is labelled by a Horn implication of the form $(X \multimap Y)$.
- (b) The root of the tree is specified as the **input** vertex. A terminal vertex, i.e. a vertex with no outgoing edges, will be specified as an **output** one.
- (c) A vertex v with exactly two outgoing edges (v, w_1) and (v, w_2) will be called **divergent**. These two outgoing edges should be labelled by Horn implications with one and the same antecedent, say $(X \multimap Y_1)$ and $(X \multimap Y_2)$, respectively.

Now, we should explain how such a program P runs for a given input W .

Definition 2.2 For a given tree-like Horn program P and any simple product W , a **strong computation** is defined by induction as follows:

We assign a simple product ²

$$\text{VALUE}(P, W, v)$$

to each vertex v of P in such a way that

- (a) For the root v_0 ,

$$\text{VALUE}(P, W, v_0) = W.$$

- (b) For every non-terminal vertex v and its son w with the edge (v, w) labelled by a Horn implication $(X \multimap Y)$, if $\text{VALUE}(P, W, v)$ is defined and, for some simple product V :

$$\text{VALUE}(P, W, v) \cong (X \otimes V),$$

then

$$\text{VALUE}(P, W, w) = (Y \otimes V).$$

Otherwise, $\text{VALUE}(P, W, w)$ is declared to be undefined.

¹ Here Σ'' represents a multiset of the form $V, \Gamma, !\Delta$, that is $\Sigma'' = V, \Gamma, !\Delta$
Recall also our convention: $(U \otimes V) = U, V$

² This $\text{VALUE}(P, W, v)$ is perceived as the intermediate value of the *strong computation* performed by P , which is obtained at point v .

Definition 2.3 For a tree-like Horn program P and a simple product W , we say that

$$P(W) = Z$$

if **for each terminal vertex w of P , $\text{VALUE}(P, W, w)$ is defined and**

$$\text{VALUE}(P, W, w) \cong Z.$$

We will describe each of our program constructs by Linear Logic formulas. Namely, we will associate a certain formula A to each edge e of a given program P , and say that

“This formula A is used on the edge e .”

Definition 2.4 Let P be a tree-like Horn program.

(a) If v is a non-divergent vertex of P with the outgoing edge e labelled by a Horn implication A , then we will say that

“Formula A itself is used on the edge e .”

(b) Let v be a divergent vertex of P with two outgoing edges e_1 and e_2 labelled by Horn implications $(X \multimap Y_1)$ and $(X \multimap Y_2)$, respectively. Then we will say that

“Formula A is used on e_1 .”

and

“Formula A is used on e_2 .”

where formula A is defined as the following \oplus -Horn implication:

$$A = (X \multimap (Y_1 \oplus Y_2)).$$

Definition 2.5 A tree-like Horn program P is said to be a **strong solution** to a Horn sequent of the form

$$W, \Delta, !\Gamma \vdash Z$$

if **for each terminal vertex w of P , $\text{VALUE}(P, W, w)$ is defined and**

$$\text{VALUE}(P, W, w) \cong Z.$$

and

(a) For every edge e in P , the formula A used on e is drawn either from Γ or from Δ .

(b) Whatever path b leading from the root to a terminal vertex we take, each formula A from Δ is used once and **exactly** once on this path b .

We prove that the Horn fragment of Linear Logic is **complete under our computational interpretation**.

Theorem 2.1 (Fairness) *Given an **HLL** derivation (with the rules from Table 2) for a Horn sequent of the form*

$$W, \Delta, !\Gamma \vdash Z,$$

we can construct a tree-like Horn program P which is a strong solution to the given sequent.

Proof. For a given **HLL** derivation, running from its leaves (axioms) to its root, we assemble a tree-like Horn program P by induction.

Below we consider all cases related to the rules from Table 2.

Case of Rule I. The elementary program from Figure 1(a), with its single vertex, will be a strong solution to any sequent of the form

$$X \vdash X.$$

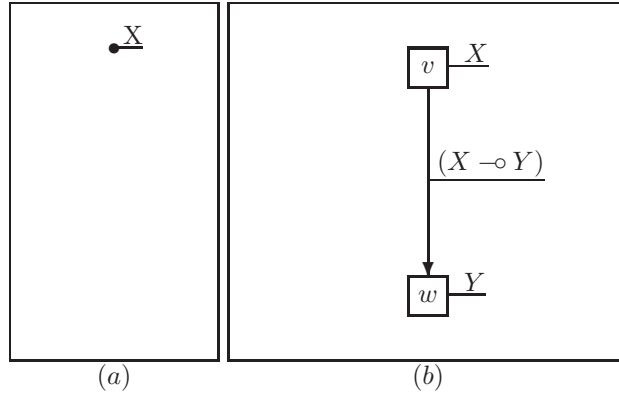


Figure 1: Elementary Horn Programs.

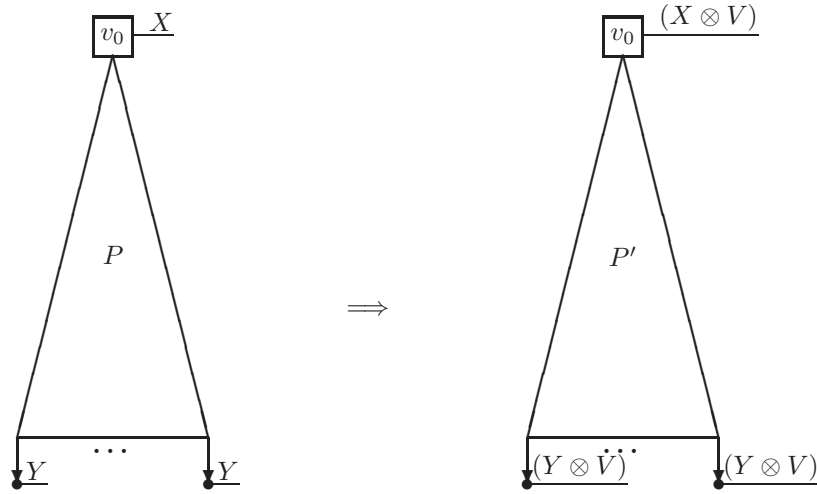


Figure 2: The "Frame property"

Case of Rule H. The elementary program consisting of a single edge labelled by $(X \multimap Y)$ (see Figure 1(b)) will be a strong solution to the sequent

$$X, (X \multimap Y) \vdash Y.$$

Case of Rule M. Suppose that P , with the input X , is a strong solution to a sequent of the form

$$X, \Gamma, !\Delta \vdash Y$$

Then as a Horn program P' we take the same P but with a larger input $(X \otimes V)$, so that, for any vertex w (see Figure 2):

$$\text{VALUE}(P', (X \otimes V), w) = (\text{VALUE}(P, X, w) \otimes V).$$

It is easily verified this Horn program P' is a strong solution to the sequent

$$(X \otimes V), \Gamma, !\Delta \vdash (Y \otimes V)$$

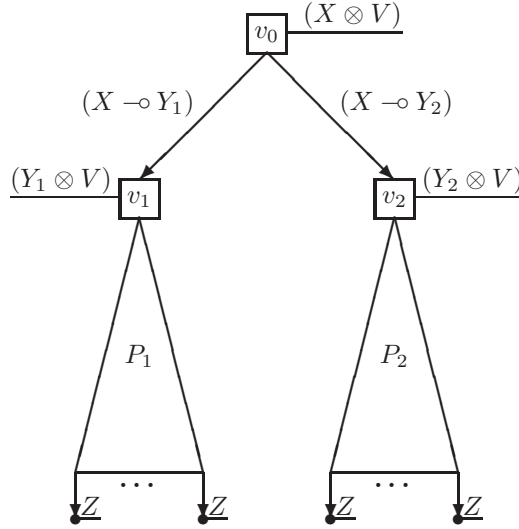


Figure 3: Strong Forking. An \oplus -Horn implication $(X \multimap (Y_1 \oplus Y_2))$ as Non-deterministic choice

Case of Rule \oplus -H. Suppose that P_1 and P_2 are strong solutions to sequents of the form

$$(Y_1 \otimes V), \Gamma, !\Delta \vdash Z$$

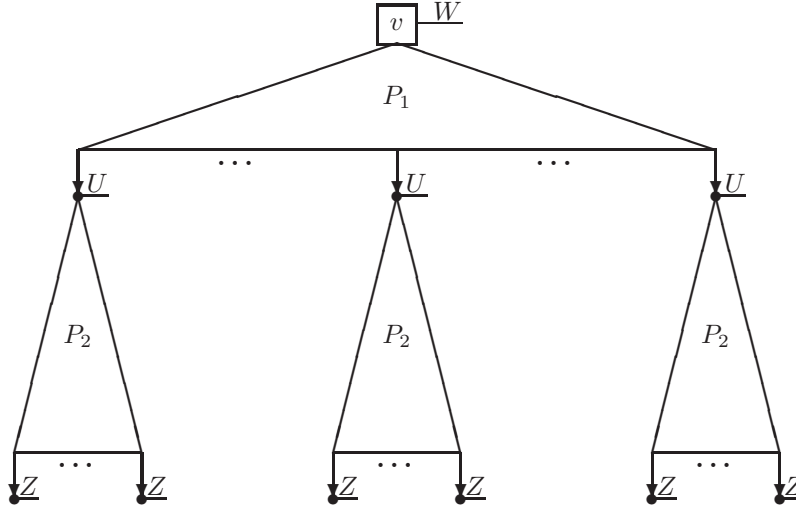
and

$$(Y_2 \otimes V), \Gamma, !\Delta \vdash Z,$$

respectively.

Now a Horn program P' can be assembled with the help of the following operation of **Strong Forking** (see Figure 3):

- (a) First, we create a new input vertex v_0 .
- (b) After that, we connect v_0 with the roots v_1 of P_1 and v_2 of P_2 , and label the edge (v_0, v_1) by the Horn implication $(X \multimap Y_1)$ and label the edge (v_0, v_2) by the Horn implication $(X \multimap Y_2)$.

Figure 4: **Cut** as Composition of programs

It is easily verified this Horn program P' is a strong solution to the sequent

$$(X \otimes V), \Gamma, (X \multimap (Y_1 \oplus Y_2)), !\Delta \vdash Z.$$

Case of Rule Cut. Suppose that P_1 and P_2 are strong solutions to sequents of the form

$$W, \Gamma_1, !\Delta_1 \vdash U$$

and

$$U, \Gamma_2, !\Delta_2 \vdash Z,$$

respectively.

Now we can construct a Horn program P' with the help of the following operation of **Composition** (see Figure 4):

(a) We glue each output vertex of P_1 to the root of a copy of the program P_2 .

It is clear that such a Horn program P' is a strong solution to the sequent

$$W, \Gamma_1, \Gamma_2, !\Delta_1, !\Delta_2 \vdash Z.$$

The rest of the Cases. Given a Horn program P_0 that is a strong solution to a sequent representing the premise for one of the remaining rules, the same Horn program P_0 can be considered as a strong solution to the corresponding conclusion sequent. ■

3 FYI: Encoding Minsky Machines

The well-known non-deterministic n -counter machines are defined as follows.

Minsky machines deal with n counters that can contain non-negative integers. The current value of an m -th counter will be represented by the variable x_m . This value

- (a) can be increased by 1, which is represented by the *assignment operation* $x_m := x_m + 1$;
- (b) or can be decreased by 1, which is represented by the *assignment operation* $x_m := x_m - 1$;

Definition 3.1 The program of an n -counter machine M is a finite list of *instructions*

$$I_1; I_2; \dots; I_s;$$

labelled by *labels*

$$L_0, L_1, L_2, \dots, L_i, \dots, L_j, \dots$$

Each of these instructions is of one of the following five types:

- (1) $L_i : x_m := x_m + 1; \textbf{goto } L_j;$
- (2) $L_i : x_m := x_m - 1; \textbf{goto } L_j;$
- (3) $L_i : \textbf{if } (x_m > 0) \textbf{ then goto } L_j;$
- (4) $L_i : \textbf{if } (x_m = 0) \textbf{ then goto } L_j;$
- (5) $L_0 : \textbf{halt};$

where L_i and L_j are *labels*, and $i \geq 1$.

Definition 3.2 An *instantaneous description (configuration)* is a tuple:

$$(L, c_1, c_2, \dots, c_n)$$

where L is a label,

c_1, c_2, \dots, c_n are the current values of our n counters, respectively.

A *computation* of a Minsky machine M is a (finite) sequence of configurations

$$K_1, K_2, \dots, K_t, K_{t+1}, \dots,$$

such that each *move* (from K_t to K_{t+1}) can be performed by applying an instruction from the program of machine M .

3.1 The Main Encoding

In our encoding we will use the following *literals*:

- (i) $r_1, r_2, \dots, r_m, \dots, r_n$ ³
- (ii) $l_0, l_1, l_2, \dots, l_i, \dots, l_j, \dots$ ⁴
- (iii) $\kappa_1, \kappa_2, \dots, \kappa_m, \dots, \kappa_n$ ⁵

Each instruction I from the list of instructions (1)-(4) of Definition 3.1 will be axiomatized by the corresponding Linear Logic formula φ_I in the following way:

$$\begin{aligned} \varphi_{(1)} &= (l_i \multimap (l_j \otimes r_m)), \\ \varphi_{(2)} &= ((l_i \otimes r_m) \multimap l_j), \\ \varphi_{(3)} &= ((l_i \otimes r_m) \multimap (l_j \otimes r_m)), \\ \varphi_{(4)} &= (l_i \multimap (l_j \oplus \kappa_m)). \end{aligned}$$

For a given machine M , its program

$$I_1; I_2; \dots; I_s;$$

is axiomatized by a multiset Φ_M as follows:

$$\Phi_M = \varphi_{I_1}, \varphi_{I_2}, \dots, \varphi_{I_s}.$$

In addition, for every m , by \mathcal{K}_m we mean the multiset consisting of one Horn implication:

$$(\kappa_m \multimap l_0)$$

³ Literal r_m is associated with the m -th counter.

⁴ Literal l_i represents label L_i .

⁵ Literal κ_m will be used to *kill* all literals except r_m .

and the following $(n - 1)$ **kill**ing implications:

$$((\kappa_m \otimes r_i) \multimap \kappa_m), \quad (i \neq m)$$

We set that

$$\mathcal{K} = \bigcup_{m=1}^n \mathcal{K}_m$$

We will prove that an **exact** correspondence exists between arbitrary computations of M on inputs

$$k_1, k_2, \dots, k_n$$

and derivations for a sequent of the form

$$(l_1 \otimes (r_1^{k_1} \otimes r_2^{k_2} \otimes \dots \otimes r_n^{k_n})), \ !\Phi_M, \ !\mathcal{K} \vdash l_0.$$

More precisely, taking into account our complete computational interpretation for sequents of this kind (Theorem 2.1), we will prove an **exact** correspondence between arbitrary computations of M on inputs

$$k_1, k_2, \dots, k_n$$

and *tree-like* strong solutions to this sequent

$$(l_1 \otimes (r_1^{k_1} \otimes r_2^{k_2} \otimes \dots \otimes r_n^{k_n})), \ !\Phi_M, \ !\mathcal{K} \vdash l_0.$$

In particular, each configuration K

$$K = (L_i, c_1, c_2, \dots, c_n)$$

will be represented in Linear Logic by a simple tensor product

$$\tilde{K} = (l_i \otimes (r_1^{c_1} \otimes r_2^{c_2} \otimes \dots \otimes r_n^{c_n})).$$

3.2 FYI: From computations to tree-like Horn programs

Lemma 3.1 *For given inputs k_1, k_2, \dots, k_n , let M be able to go from the initial configuration $(L_1, k_1, k_2, \dots, k_n)$ to the halting configuration $(L_0, 0, 0, \dots, 0)$.*

Then we can construct a tree-like Horn program P , which is a strong solution to the sequent

$$(l_1 \otimes (r_1^{k_1} \otimes r_2^{k_2} \otimes \dots \otimes r_n^{k_n})), \ !\Phi_M, \ !\mathcal{K} \vdash l_0$$

Proof. Let

$$K_0, K_1, K_2, \dots, K_u, K_{u+1}, \dots, K_t$$

be a computation of M (See Figure 5(a)) leading from the initial configuration K_0 :

$$K_0 = (L_1, k_1, k_2, \dots, k_n),$$

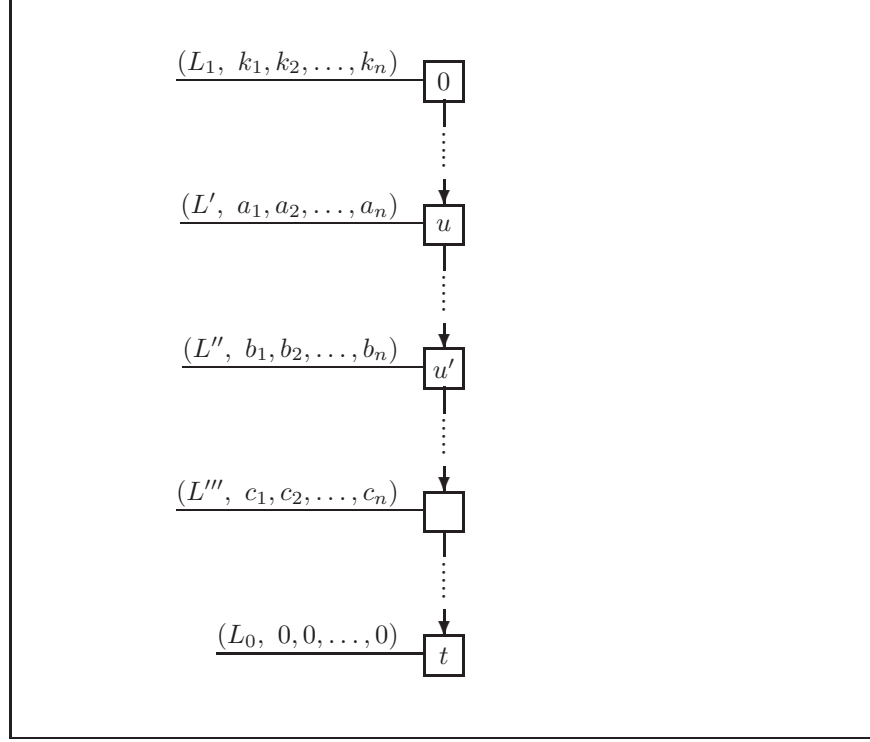
to the halting configuration K_t :

$$K_t = (L_0, 0, 0, \dots, 0).$$

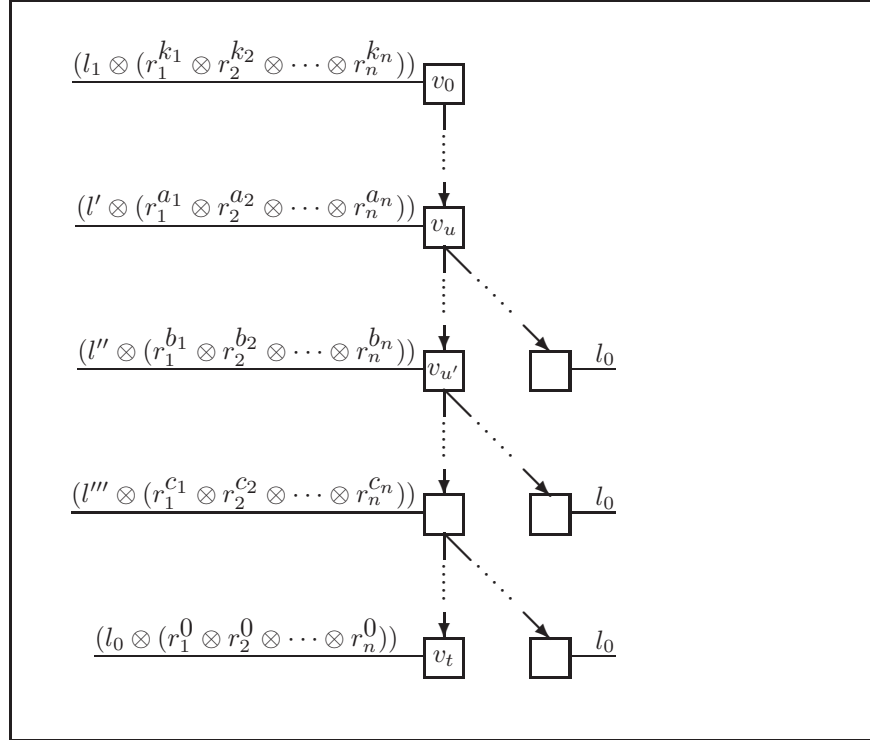
Running from the beginning of this sequence of configurations to its end, we will construct a *tree-like* Horn program P , which is a strong solution to the sequent

$$(l_1 \otimes (r_1^{k_1} \otimes r_2^{k_2} \otimes \dots \otimes r_n^{k_n})), \ !\Phi_M, \ !\mathcal{K} \vdash l_0,$$

and has the following peculiarities (See Figure 5(b))



(a)



(b)

Figure 5: The correspondence: Computation (a) — Horn Program (b).

- (i) $P(\widetilde{K}_0) = \widetilde{K}_t = l_0$,
- (ii) and, moreover, there exists a branch of P , we call it *main*:

$$v_0, v_1, v_2, \dots, v_u, v_{u+1}, \dots, v_t,$$

such that for each vertex v_u from this *main* branch:

$$\text{VALUE}(P, \widetilde{K}_0, v_u) \cong \widetilde{K}_u.$$

We construct the desired program P by induction:

The root v_0 of P is associated with the initial configuration K_0 :

$$\text{VALUE}(P, \widetilde{K}_0, v_0) = (l_1 \otimes (r_1^{k_1} \otimes r_2^{k_2} \otimes \dots \otimes r_n^{k_n})).$$

Let v_u be the terminal vertex of the fragment of P (that has already been constructed), associated with the current configuration K_u :

$$\text{VALUE}(P, \widetilde{K}_0, v_u) \cong \widetilde{K}_u = (l_i \otimes (r_1^{a_1} \otimes r_2^{a_2} \otimes \dots \otimes r_n^{a_n})).$$

The move from K_u to K_{u+1} is simulated in the following way:

- (a) If this move is performed by applying an *assignment operation* instruction I from the list of instructions (1)-(3) of Definition 3.1, then we create a new edge (v_u, v_{u+1}) and label this new edge by the corresponding Horn implication φ_I , getting for this new terminal vertex v_{u+1} :

$$\text{VALUE}(P, \widetilde{K}_0, v_{u+1}) \cong \widetilde{K}_{u+1}.$$

Figure 6 shows the case where this instruction I is of the form $L_i : x_1 := x_1 - 1$; **goto** L_j .

- (b) Let the foregoing move be performed by applying a *ZERO-test* instruction I of the form (4)

$$L_i : \text{if } (x_m = 0) \text{ then goto } L_j.$$

The definability conditions of this move provide that

$$a_m = 0.$$

We extend the fragment of P (that has already been constructed) as follows (See Figure 7):

First, we create two new outgoing edges (v_u, v_{u+1}) and (v_u, w_u) , and label these new edges by the Horn implications

$$(l_i \multimap l_j) \text{ and } (l_i \multimap \kappa_m),$$

respectively. It is readily seen that

$$\begin{aligned} \text{VALUE}(P, \widetilde{K}_0, v_{u+1}) &\cong (l_j \otimes (r_1^{a_1} \otimes r_2^{a_2} \otimes \dots \otimes r_m^{a_m} \otimes \dots \otimes r_n^{a_n})) = \widetilde{K}_{u+1}, \\ \text{VALUE}(P, \widetilde{K}_0, w_u) &\cong (\kappa_m \otimes (r_1^{a_1} \otimes r_2^{a_2} \otimes \dots \otimes r_m^{a_m} \otimes \dots \otimes r_n^{a_n})). \end{aligned}$$

Then, we create a chain of t_u new edges

$$(w_u, w_1^u), (w_1^u, w_2^u), \dots, (w_{t_u}^u - 1, w_{t_u}^u)$$

where

$$t_u = a_1 + a_2 + \dots + a_{m-1} + a_{m+1} + \dots + a_n,$$

and label these new edges by such Horn implications from \mathcal{K}_m as to *kill* all occurrences of literals

$$r_1, r_2, \dots, r_{m-1}, r_{m+1}, \dots, r_n,$$

and ensure that

$$\text{VALUE}(P, \widetilde{K}_0, w_{t_u}^u) \cong (\kappa_m \otimes (r_1^0 \otimes r_2^0 \otimes \cdots \otimes r_m^{a_m} \otimes \cdots \otimes r_n^0)).$$

Finally, we create a new edge $(w_{t_u}^u, w_{t_u}^u + 1)$, and label this new edge by the Horn implication

$$(\kappa_m \multimap l_0).$$

Taking into account that $a_m = 0$, for the terminal vertex $w_{t_u}^u + 1$ of the foregoing chain, we have:

$$\text{VALUE}(P, \widetilde{K}_0, w_{t_u}^u + 1) = (l_0 \otimes (r_1^0 \otimes r_2^0 \otimes \cdots \otimes r_m^{a_m} \otimes \cdots \otimes r_n^0)) = l_0.$$

Hence, for all terminal vertices w , i.e. both for the terminal vertex v_t of the *main* branch and for the terminal vertices of all auxiliary chains, we obtain that

$$\text{VALUE}(P, \widetilde{K}_0, w) = l_0 = \widetilde{K}_t.$$

Thus, our inductive process results in a *tree-like* Horn program P that is a strong solution to the sequent

$$(l_1 \otimes (r_1^{k_1} \otimes r_2^{k_2} \otimes \cdots \otimes r_n^{k_n})), !\Phi_M, !\mathcal{K} \vdash l_0$$

■

4 From tree-like Horn programs to Minsky computations

Theorem 4.1 *For given integers k_1, k_2, \dots, k_n , let a sequent of the form*

$$(l_1 \otimes (r_1^{k_1} \otimes r_2^{k_2} \otimes \cdots \otimes r_n^{k_n})), !\Phi_M, !\mathcal{K} \vdash l_0$$

be derivable in Linear Logic. Then M can go from an initial configuration of the form

$$(L_1, k_1, k_2, \dots, k_n)$$

to the halting configuration

$$(L_0, 0, 0, \dots, 0).$$

Proof. By Theorem 1.1 and Theorem 2.1, we can construct a *tree-like* Horn program P such that

- (i) Each of the Horn implications occurring in P is drawn either from Φ_M or from \mathcal{K} .
- (ii) **For all terminal vertices w of P :**

$$\text{VALUE}(P, W_0, w) = l_0$$

where

$$W_0 = (l_1 \otimes (r_1^{k_1} \otimes r_2^{k_2} \otimes \cdots \otimes r_n^{k_n})).$$

Lemma 4.1 *Running from the root v_0 to terminal vertices of P , we assemble the desired Minsky computation as follows:*

- (a) *It is proved that program P cannot be but of the form represented in Figure 5.*
- (b) *We can identify a branch of P , called the main branch:*

$$v_0, v_1, v_2, \dots, v_u, v_{u+1}, \dots, v_t,$$

such that for all vertices v_u on this branch, $\text{VALUE}(P, W_0, v_u)$ is proved to be of the form

$$\text{VALUE}(P, W_0, v_u) \cong (l_i \otimes (r_1^{a_1} \otimes r_2^{a_2} \otimes \cdots \otimes r_n^{a_n})).$$

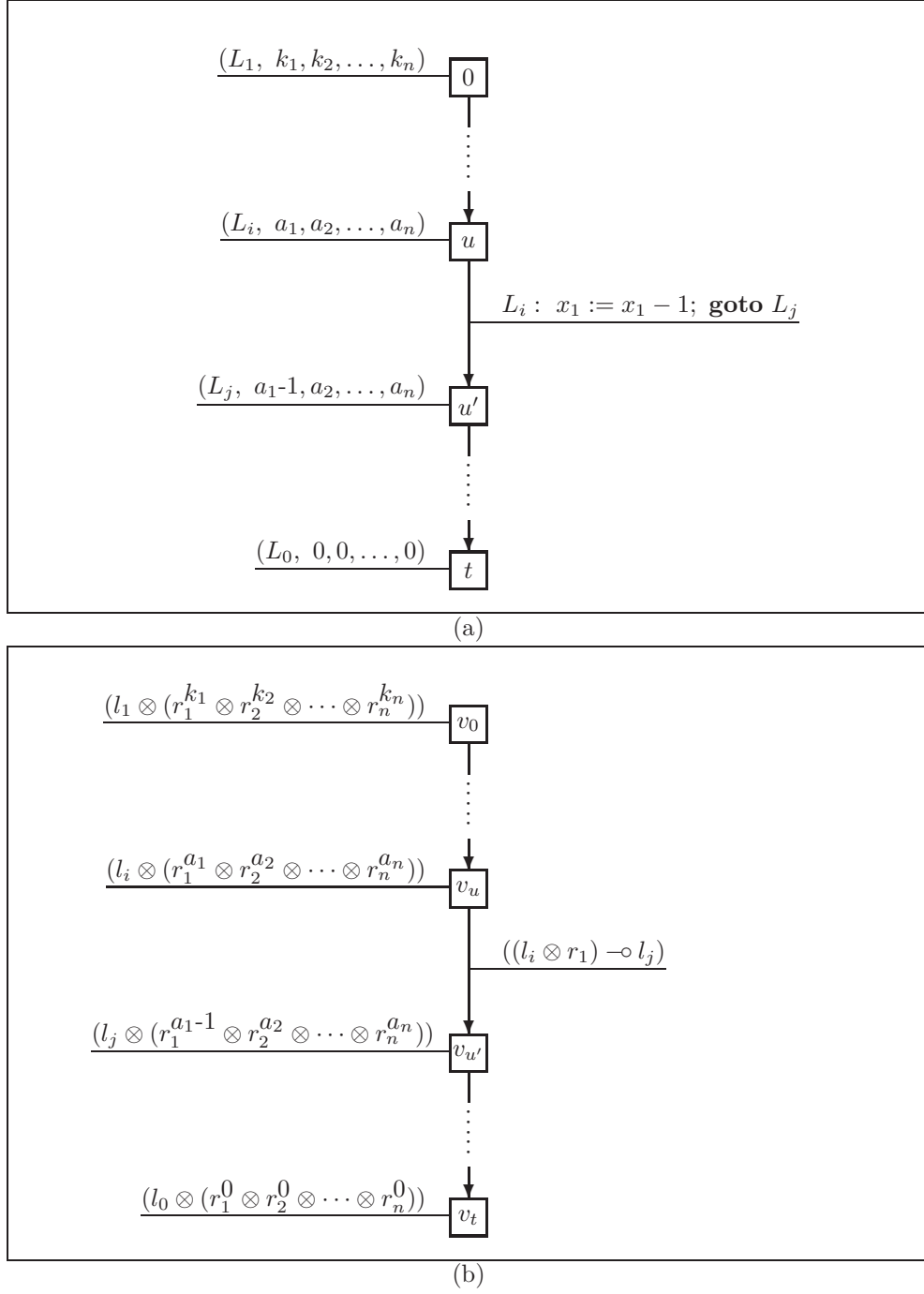
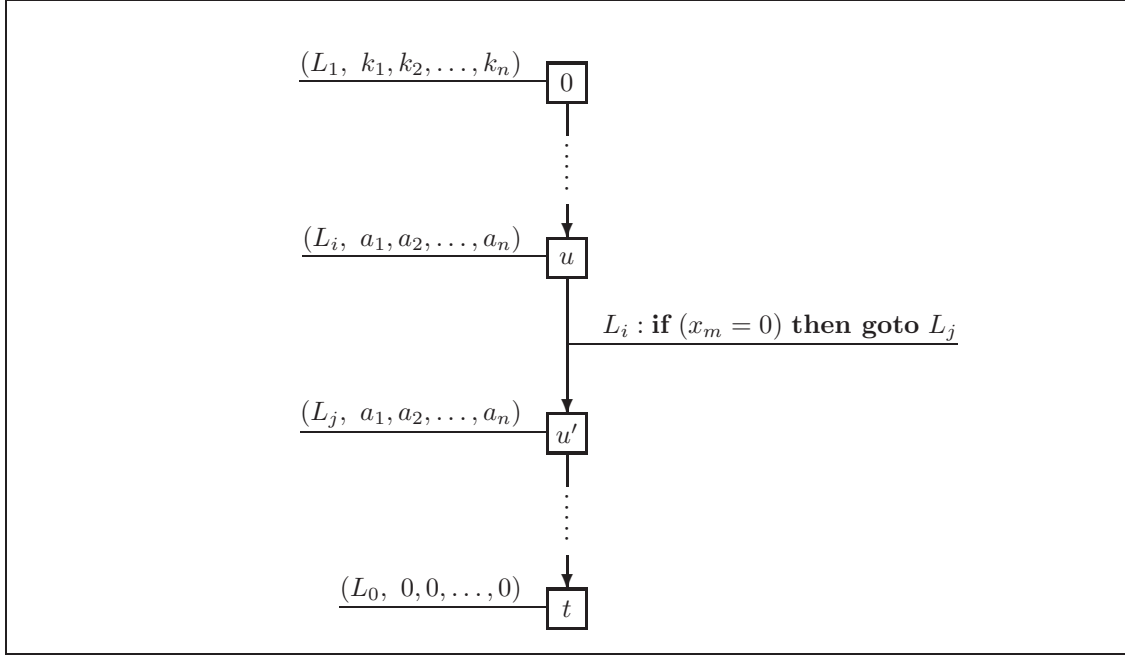
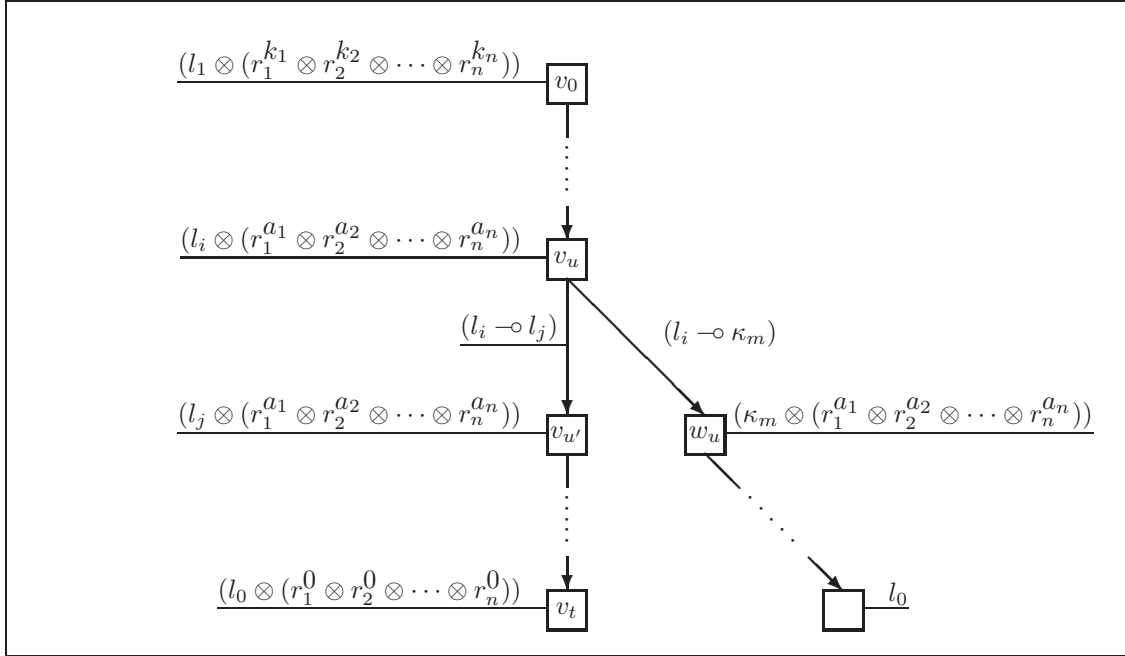


Figure 6: The *assignment operation* correspondence: (a) – (b).



(a)



(b)

Figure 7: The *ZERO-test* correspondence: (a) – (b).

- (c) For all non-terminal vertices w' of P that are outside the main branch, $\text{VALUE}(P, W_0, w')$ is proved to be of the form

$$\text{VALUE}(P, W_0, w') \cong (\kappa_m \otimes (r_1^{a_1} \otimes r_2^{a_2} \otimes \cdots \otimes r_n^{a_n})).$$

- (d) Finally, the following sequence of configurations (See Figure 5)

$$K_0, K_1, K_2, \dots, K_u, K_{u+1}, \dots, K_t$$

such that for every integer u

$$\widetilde{K}_u \cong \text{VALUE}(P, W_0, v_u),$$

is proved to be a successful computation of M leading from the initial configuration K_0 :

$$K_0 = (L_1, k_1, k_2, \dots, k_n),$$

to the halting configuration K_t :

$$K_t = (L_0, 0, 0, \dots, 0).$$

Proof. Since

$$\widetilde{K}_0 = \text{VALUE}(P, W_0, v_0) = (l_1 \otimes (r_1^{k_1} \otimes r_2^{k_2} \otimes \cdots \otimes r_n^{k_n})),$$

we have for the root v_0 :

$$K_0 = (L_1, k_1, k_2, \dots, k_n).$$

Let v_u be the current vertex on the *main* branch we are searching for, and, according to the inductive hypothesis, let $\text{VALUE}(P, W_0, v_u)$ be of the form

$$\text{VALUE}(P, W_0, v_u) \cong \widetilde{K}_u = (l_i \otimes (r_1^{a_1} \otimes r_2^{a_2} \otimes \cdots \otimes r_n^{a_n})).$$

There are the following cases to be considered.

- (a) Suppose that v_u is a non-divergent vertex with the only son which will be named v_{u+1} . According to the definability conditions for our program P , the single outgoing edge (v_u, v_{u+1}) cannot be labelled but by a Horn implication A from Φ_M . Moreover,

$$A = \varphi_I$$

for some instruction I of the form (1)-(3) from Definition 3.1. Let this instruction I be of the form

$$L_i : x_1 := x_1 - 1; \text{ goto } L_j;$$

and

$$A = ((l_i \otimes r_1) \multimap l_j).$$

Then we have (See Figure 6):

$$\text{VALUE}(P, W_0, v_{u+1}) \cong (l_j \otimes (r_1^{a_1-1} \otimes r_2^{a_2} \otimes \cdots \otimes r_n^{a_n})),$$

(and, hence, $a_1 \geq 1$). Performing the foregoing instruction I , machine M can move from the current configuration K_u :

$$K_u = (L_i, a_1, a_2, \dots, a_n),$$

to the next configuration K_{u+1} :

$$K_{u+1} = (L_j, a_1-1, a_2, \dots, a_n),$$

such that

$$\widetilde{K}_{u+1} \cong \text{VALUE}(P, W_0, v_{u+1}).$$

The remaining cases are handled similarly.

- (b) The **crucial point** is where v_u is a vertex with two outgoing edges, say (v_u, v_{u+1}) and (v_u, w_u) , labelled¹⁹ by Horn implications A_1 and A_2 , respectively. (See Figure 7)

It means that the implication used at this point of program P must be a \oplus -Horn implication A from Φ_M of the form

$$A = (l_i \multimap (l_j \oplus \kappa_m)),$$

and, in addition,

$$\begin{aligned} A_1 &= (l_i \multimap l_j) \\ A_2 &= (l_i \multimap \kappa_m). \end{aligned}$$

Therefore,

$$\begin{aligned} \text{VALUE}(P, W_0, v_{u+1}) &\cong (l_j \otimes (r_1^{a_1} \otimes r_2^{a_2} \otimes \cdots \otimes r_m^{a_m} \otimes \cdots \otimes r_n^{a_n})) \\ \text{VALUE}(P, W_0, w_u) &\cong (\kappa_m \otimes (r_1^{a_1} \otimes r_2^{a_2} \otimes \cdots \otimes r_m^{a_m} \otimes \cdots \otimes r_n^{a_n})). \end{aligned}$$

Let us examine the descendants of the vertex w_u .

Taking into account the definability conditions, any edge (w_1, w_2) , such that w_1 is a descendant of w_u , cannot be labelled but by a Horn implication from \mathcal{K}_m .

So we can conclude that for all non-terminal descendants w' of the vertex w_u , $\text{VALUE}(P, W_0, w')$ is of the form

$$\text{VALUE}(P, W_0, w') \cong (\kappa_m \otimes (r_1^{c_1} \otimes r_2^{c_2} \otimes \cdots \otimes r_m^{a_m} \otimes \cdots \otimes r_n^{c_n})).$$

For the terminal descendant w of the vertex w_u , $\text{VALUE}(P, W_0, w)$ is to be of the form

$$\text{VALUE}(P, W_0, w) \cong (l_0 \otimes (r_1^{c_1} \otimes r_2^{c_2} \otimes \cdots \otimes r_m^{a_m} \otimes \cdots \otimes r_n^{c_n})).$$

Recalling that

$$\text{VALUE}(P, W_0, w) = l_0,$$

we get the desired

$$a_m = 0.$$

Indeed,

$$A = \varphi_I$$

for a *ZERO-test* instruction I of the form

$$L_i : \text{ if } (x_m = 0) \text{ then goto } L_j;$$

Performing this instruction I , machine M can move from the current configuration K_u :

$$K_u = (L_i, a_1, a_2, \dots, a_m, \dots, a_n),$$

to the next configuration K_{u+1} :

$$K_{u+1} = (L_j, a_1, a_2, \dots, a_m, \dots, a_n),$$

such that

$$\widetilde{K_{u+1}} \cong \text{VALUE}(P, W_0, v_{u+1}).$$

- (c) Suppose that the *main* branch we have been developing

$$v_0, v_1, v_2, \dots, v_u, v_{u+1}, \dots, v_t,$$

has ended at a vertex v_t . According to what has been said,

$$l_0 = \text{VALUE}(P, W_0, v_t) \cong \widetilde{K_t} = (l_j \otimes (r_1^{c_1} \otimes r_2^{c_2} \otimes \cdots \otimes r_n^{c_n})).$$

Hence, this configuration K_t is the *halting* one:

$$K_t = (L_0, 0, 0, \dots, 0).$$

Now, bringing together all the cases considered, we can complete Lemma 4.1 and, hence, Theorem 4.1. 20

Theorem 4.2 *For given inputs k_1, k_2, \dots, k_n , an n -counter Minsky machine M can go from the initial configuration $(L_1, k_1, k_2, \dots, k_n)$ to the halting configuration $(L_0, 0, 0, \dots, 0)$ if and only if a sequent of the form*

$$(l_1 \otimes (r_1^{k_1} \otimes r_2^{k_2} \otimes \dots \otimes r_n^{k_n})), !\Phi_M, !\mathcal{K} \vdash l_0$$

is derivable in Linear Logic.

Proof. We bring together Lemma 3.1 and Theorem 4.1. ■

References

- [1] M.R.Garey and D.S.Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. 1979.
- [2] V.Gehlot and C.A.Gunter. Normal process representatives. In *Proc. 5-th Annual IEEE Symposium on Logic in Computer Science, Philadelphia*, June 1990.
- [3] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1, 1987, pp.1-102.
- [4] Jean-Yves Girard. Linear logic: its syntax and semantics, In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, London Mathematical Society Lecture Notes, Vol. 222, pp.1-42. Cambridge University Press, 1995.
- [5] Max Kanovich. Horn Programming in Linear Logic is NP-complete. In *Proc. 7-th Annual IEEE Symposium on Logic in Computer Science, Santa Cruz*, June 1992, pp.200-210
- [6] Max Kanovich. Linear logic as a logic of computations, *Annals of Pure and Applied Logic*, 67 (1994) pp.183-212
- [7] Max Kanovich. The complexity of Horn fragments of linear logic. *Annals of Pure and Applied Logic*, 69:195–241, 1994.
- [8] Max Kanovich. The Direct Simulation of Minsky machines in Linear logic, In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, London Mathematical Society Lecture Notes, Vol. 222, pp.123-145. Cambridge University Press, 1995.
- [9] Max Kanovich. Petri Nets, Horn Programs, Linear Logic, and Vector Games, *Annals of Pure and Applied Logic*, 75 (1995) pp.107-135
- [10] R.S.Kosaraju. Decidability of reachability in vector addition systems. In *Proc. 14-th Annual Symposium on Theory of Computing*, 267-281, 1982. Preliminary Version.
- [11] P.Lincoln, J.Mitchell, A.Scedrov, and N.Shankar. Decision Problems for Propositional Linear Logic. *Annals of Pure and Applied Logic*, 56 (1992) pp.239-311.
- [12] Patrick Lincoln. Deciding Provability of Linear Logic Formulas, In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, London Mathematical Society Lecture Notes, Vol. 222, pp.109-122. Cambridge University Press, 1995.
- [13] N.Marti-Oliet and J.Meseguer. From Petri Nets to Linear Logic. In: Springer LNCS 389, ed. by D.R.Pitt et al., 1989, pp.313-340
- [14] E.Mayr, An algorithm for the general Petri net reachability problem. *SIAM J.Comput.*, 13, N 3, 441-460, 1984.
- [15] M.Minsky. Recursive unsolvability of Post's problem of 'tag' and other topics in the theory of Turing machines. *Annals of Mathematics*, 74:3:437-455, 1961.